Downgrading DNSSEC: How to Exploit Crypto Agility for Hijacking Signed Zones

Elias Heftrig, *ATHENE and Fraunhofer SIT;* Haya Shulman, *ATHENE, Fraunhofer SIT, and Goethe-Universität Frankfurt;* Michael Waidner, *ATHENE, Fraunhofer SIT, and Technische Universität Darmstadt*

https://www.usenix.org/conference/usenixsecurity23/presentation/heftrig

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

Downgrading DNSSEC: How to Exploit Crypto Agility for Hijacking Signed Zones

Elias Heftrig
ATHENE
Fraunhofer SIT

Haya Shulman
ATHENE
Fraunhofer SIT
Goethe-Universität Frankfurt

Michael Waidner
ATHENE
Fraunhofer SIT
Technische Universität Darmstadt

Abstract

Cryptographic algorithm agility is an important property for DNSSEC: it allows easy deployment of new algorithms if the existing ones are no longer secure. Significant operational and research efforts are dedicated to pushing the deployment of new algorithms in DNSSEC forward. Recent research shows that DNSSEC is gradually achieving algorithm agility: most DNSSEC supporting resolvers can validate a number of different algorithms and domains are increasingly signed with cryptographically strong ciphers.

In this work we show for the first time that the cryptographic agility in DNSSEC, although critical for making DNS secure with strong cryptography, also introduces a severe vulnerability. We find that under certain conditions, when new, unsupported algorithms are listed in signed DNS responses, the resolvers do not validate DNSSEC. As a result, domains that deploy new ciphers, risk exposing the validating resolvers to cache poisoning attacks. We use this to develop DNSSEC-downgrade attacks and experimentally and ethically evaluate our attacks against popular DNS resolver implementations, public DNS providers, and DNS resolvers used by web clients.

We validate the success of DNSSEC-downgrade attacks by poisoning the resolvers: we inject fake records, in signed domains, into the caches of validating resolvers. Our evaluations showed that during 2021 major DNS providers, such as Google Public DNS and Cloudflare, as well as 35% of DNS resolvers used by the web clients were vulnerable to our attacks. After coordinated disclosure with the affected operators, that number reduced to 5.03% in 2022.

We trace the factors that led to this situation and provide recommendations.

1 Introduction

DNSSEC [RFC4033-RFC4035] was designed to prevent DNS cache poisoning attacks. Proposed and standardized in the 90s, DNSSEC is slowly gaining traction and increasingly more networks are now supporting DNSSEC. Our measurements from November 2022 indicate that 5.26% of the domains on 1M-Tranco list are signed and 31.02% of the Internet clients use DNSSEC-validating DNS resolvers. These measurements show that the deployment of DNSSEC has not substantially changed after our initial measurements in 2021. Our findings are also inline with the measurements of signed zones by SecSpider[1] and measurements of validating resolvers by APNIC[2], whose longitudinal data collection, except occasional spikes, exhibit a slight decline in DNSSEC support between 2021 and 2022. Although the number of DNSSEC-validating resolvers tripled since 2014, most resolvers still support a limited set of algorithms and many domains are signed with algorithms that are no longer considered secure. Unfortunately, replacing existing or adding new ciphers to DNSSEC is challenging.

Cryptographic algorithms agility. The research and operational communities invest significant efforts to measure the currently supported ciphers, as well as to explore obstacles towards deployment of new ciphers in DNSSEC. Initially, the DNSSEC standard allowed domains to use either DSA/SHA1 or RSA/SHA1 for signing their zones [RFC4034] - these are no longer deemed secure. Since then, additional algorithms were included in DNSSEC [RFC5155,5702,5933,6605,8080]. The domain owners can now use any subset of 13 algorithms for signing their zones [1]. Although it still takes long to standardize new algorithms and remove deprecated ones, it is generally believed that DNSSEC has partially achieved algorithm agility [29]. However, no work has been carried out to understand the implications of deployment of new algorithms on the security of DNS.

We show that the current state of algorithm agility in DNSSEC introduces a vulnerability, we demonstrate how to exploit it to downgrade DNSSEC. Our analysis of DNSSEC RFCs and DNS resolvers' behaviour indicates that lack of clear recommendations for handling new ciphers is the main factor leading to the vulnerabilities.

Unclear specifications for handling unknown ciphers. According to DNSSEC standard, when returning a lookup

[1] https://secspider.net/stats.html
[2] https://stats.labs.apnic.net/dnssec/

result in a signed zone a DNSSEC supporting resolver should either return correctly validated records signalling authenticated data, or should return SERVFAIL when the data cannot be authenticated. However, the DNSSEC standard does not clearly specify the recommended behaviour for DNS resolvers when faced with new ciphers. Should the resolvers accept records that are signed with unknown algorithms or reject them? How should the validation proceed when a domain supports multiple algorithms, only some of which are unknown? How should the resolvers react in case of inconsistencies in keys between the parent and the child zones? We experimentally show that this lack of clear specification in the DNSSEC standard leads to different vulnerable behaviour implementations at the resolvers: in presence of unknown algorithms in DNSSEC records the resolvers accept the records in the responses without validating them. Even if the signatures are invalid or if a chain of trust cannot be established to the root, some resolvers do not return SERVFAIL, but instead accept the DNS records without validation.

Adding new ciphers can disable DNSSEC validation. In this work we find that domains that adopt new ciphers, e.g., ED448 [RFC8080], under some conditions risk disabling the DNSSEC validation of DNS resolvers that do not support the new ciphers. The cause is a lack of clarity and rigor in defining and implementing DNSSEC for new ciphers. While some public DNS providers, such as Cisco Umbrella OpenDNS, already support ED448, most DNS implementations and DNS operators still do not. In our measurements of resolvers we discover that even large public DNS providers in such cases may not validate DNSSEC.

DNSSEC downgrade attacks. More significantly, we show that resolvers may be vulnerable to *downgrade attacks* even for zones that are signed with widely known algorithms, such as RSASHA1. The idea behind our attacks is to manipulate the algorithm numbers in DNSSEC records or to remove DNSSEC records, e.g., DNSKEY, DS, RRSIG records. This causes DNSSEC-supporting resolvers not to apply DNSSEC validation over DNS records, and exposes them to cache poisoning attacks.

Factors exposing to vulnerabilities. The gaps in the standard DNSSEC specifications that expose to vulnerabilities that we exploit in this work can be summarized as follows:

Attack surface: The DNSSEC RFCs leave an attack surface which allows stripping off RRSIGs, without being detected by the validating resolvers, as in [RFC6840].

Definition of security states: The definition of the security states from the DNSSEC validation is ambiguous across the different RFCs. For instance, the way "indeterminate" and "bogus" validation states are defined in [RFC4033] and [RFC4035]. The DNSSEC term "bogus" describes records that are invalid. On the other hand, 'indeterminate' is defined as the security state of data that could not be authenticated, because no trust anchor has been configured for the subtree of DNS, which the records to be authenticated are located in. This requires the resolver to have a trust anchor at some node other than the root – but not at the root itself – which is not common. The definitions result in different interpretations by the developers leading to discrepancies in behaviour between the different resolver implementations. Re-definition of states causes follow-up specifications to try and reconcile differing interpretations. One example is [RFC7672] which defines a security scheme, that applies DNSSEC to improve e-mail security ("Note that the "indeterminate" security status has a conflicting definition in Section 5 of [RFC4033]. [...] In this document, the term "indeterminate" will be used exclusively in the [RFC4035] sense.").

Definition of validity: There are repeated statements and updates to those across different RFCs. This considerably increases opportunity for interpretation errors (and also increases opportunity for errors in specification updates). In some cases updated definitions expose to vulnerabilities, such as [RFC6840] watering down resolver expectations from [RFC4035].

The role of standards. The standards typically aim and need to offer some flexibility of implementation in order to allow for competition among products meeting the standard. On the other hand, it is important to avoid standards that wind up allowing implementations that defeat the purpose of the standard. In this context an interesting question is what kind of analysis of new standards should be undertaken before they are adopted and how implementations should be evaluated against the standard. These are fundamental issues raised by the vulnerabilities found in our, as well as prior, works, e.g., [12, 25]. Therefore, our work raises the question of how we solve not only this particular problem with this particular standard, but also the more general problem for existing and future standards.

Ethics and disclosure. We took preliminary steps to address the vulnerabilities found in this work by contacting the DNS software vendors and public DNS providers. We contacted the following DNS vendors and operators: Nic.fr, Verisign, NLnetlabs, Cloudflare and Google were notified on September 2021. Microsoft and OpenDNS in July 2022. Details of the notifications are in Appendix, Section A.

We experimentally evaluated the attacks reported in this work against servers that we set up as well as against open DNS resolvers and public DNS resolvers, and against resolvers of web clients in the Internet. In our evaluations we used domains that we control. This allowed us to validate the downgrade attacks without downgrading the DNSSEC-security of real domains. To collect information about vulnerabilities in client side DNS equipment, we use an ad network. Previous measurements of DNSSEC used ad network to infer information about the client side DNS resolvers [22, 23]. Such studies however involve ethical concerns: they may introduce load, expose privacy of a user or its behaviour, may expose users to objectionable content or may load resources from domains that may be legally risky. In our study and data

collection we follow the ethical guidelines for network measurements defined in [34], which were also approved by the ethical committee of our scientific organization. By following these ethical guidelines we ensure that the clients equipment is not overloaded, the privacy of the clients is preserved, the clients are not exposed to security risks, are not redirected to any other domain or host, and no client is presented insecure or objectable content.

• *Minimal load:* to minimise load on clients we reduce significantly the number of queries we trigger from the stub resolvers to the recursive resolvers . During the ad-net evaluations the pop-under carrying our investigation script takes only 150KB on wire. The loaded web resources (images; up to 180 per client) each have a negligible size of mere 84B on wire. Afterwards the communication is performed against the recursive resolvers with the average of 3 queries per resolver. The volume of the packets we exchange with the recursive resolvers do not exceed those in other studies of DNS, e.g., [25]. Our tests are designed to use a single referral response from our nameserver and we also introduce up to 50ms randomized delay between the queries to the recursive resolvers. Benchmark studies[3] show that modern Knot DNS server is able to serve 500K DNS packets per second. Even older benchmark studies [41] in 2009 on performance of validating resolvers with commodity hardware found that Bind and Unbound could handle more than 1K queries per second with signed zones. Hence, the additional DNS requests volume of 3 queries per second triggered by our experiments do not load the recursive resolvers.

• *User consent:* from the users' perspective, our measurements over an ad-network are simply advertisements and third party content that are loaded when visiting web sites. However, we do not experiment on the users themselves, since our experiment is on the network equipment between the users, their resolvers and our nameservers, which is independent of the users and their behaviour.

• *Privacy:* the data we collect is processed and analyzed during the communication with the recursive resolvers. We do not collect and do not store on our server any user sensitive or personal data, but only the statistics of the analysis. The stored data contains a timestamp of the request, a randomly generated token associated with the resolver, and query URL. We do not study user behaviour and do not collect any data related to user behaviour. Our online service provides a privacy policy and displays only information about the computed statistics, ensuring data and source privacy.

• *User security:* our measurements infrastructure triggers requests only to the nameservers that we set up and control. Hence, no client was redirected by our measurements to any other domain or host, and no user was presented insecure or objectable content.

Contributions. We make the following contributions:

[3] https://www.knot-dns.cz/benchmark/

• We systematically analyze the different conditions under which DNS resolvers can be forced to skip DNSSEC validation and develop methodologies for evaluating the DNSSEC validation in DNS resolvers. We find that the validation logic in popular DNS implementations and in public DNS resolvers is often flawed. We develop attacks, including for injecting adversary's keys into the victim resolver's cache for a secure zone and for disabling a validation for a secure zone.

• In our study during 2021 we found that some major DNS providers, such as Google Public DNS and Cloudflare, were vulnerable to our downgrade attacks. Over the course of our research, we also found that 65% of open resolvers and almost 35% of the resolvers used by web clients were vulnerable to downgrade attacks. After going through responsible disclosure process with the affected operators, our measurements in November 2022 show these numbers reduced down to 7.50% of open resolvers and 5% of resolvers used by web clients.

• We explore key factors causing the vulnerabilities and provide recommendations for preventing our DNSSEC downgrade attacks. We develop a tool for testing vulnerable DNSSEC behaviour: `https://www.dnssec-downgrade.net/`

Organisation. We review related work and provide an overview of DNS security in Section 2 and background on DNSSEC in Section 3. In Section 4 we introduce the datasets of resolvers and domains that we study in this work, and provide our measurements of the deployment of DNSSEC. In Section 5 we describe our DNSSEC downgrade methodology and evaluate the attacks in Section 6. We discuss countermeasures in Section 7 and conclude this work in Section 8.

2 DNS Cache Poisoning and Defences

Domain Name System (DNS) [28] cache poisoning is an attack in which an adversary injects malicious records into a victim's DNS cache, to hijack a victim domain. Such attacks are especially effective against caching DNS resolvers. Once the attack is successful, the fake records are cached, subsequent requests for the poisoned resource are responded to with the malicious value from the cache, redirecting all the clients of the compromised resolver to an adversarial host. The attacker can intercept the traffic between the services (such as web, email, FTP) in the victim domain and the hosts that use the poisoned cache. SSL/TLS would prevent such attacks, since the redirection would cause a certificate error. Nevertheless, DNS cache poisoning can also be exploited to issue fraudulent certificates, in which case no error messages would be issued and even security savvy users may not detect the attacks [4, 5, 8]. To make cache poisoning attacks difficult to launch, defences and best practices were developed.

DNS cache poisoning chronicle. DNS experts have been warning for over two decades that source ports and Transaction Identifiers (TXID) have to be sufficiently random to make DNS cache poisoning attacks impractical. Vixie recom-

mended to randomise the source ports already in 1995 [39] and Bernstein in 2002 [3]. In 2007 Klein identified vulnerability in Bind9 [17] and in Windows DNS [18] resolvers allowing to reduce the entropy in the TXID. In 2008 Kaminsky [16] presented a practical cache poisoning attack even against truly randomised TXID, by generating multiple DNS responses, each with a different TXID value. Following the Kaminsky attack, DNS resolvers were patched against cache poisoning [14], and most resolvers randomised the UDP source ports in queries. Despite multiple efforts to randomise the ports, every improvement was often met with a new derandomisation technique. In 2012 [10] developed side channels to infer the source ports in DNS requests. The attacks targeted DNS resolvers located behind NAT devices. [11, 20, 35, 43] exploit fragmentation to inject spoofed records into DNS responses in different setups against caching resolvers and forwarders. A followup work demonstrated effectiveness of such cache poisoning attacks also against stub resolvers [2]. Side channels were also used to predict the ports due to vulnerable PRNG in Linux kernel [19], these are however difficult to apply in practice. [25] developed a method to leverage ICMP errors to infer the UDP source ports selected by DNS resolvers. The attacker exploits a global ICMP rate limit, which leaks information about the selected UDP port. A recent work [26] improved an ICMP based side channel, developing an attack that uses ICMP probes to infer a source UDP port.

Defences against on-path poisoning. Cryptographic signatures with DNSSEC [RFC6840] [40] aim to prevent on-path attacks: the domain signs the records, and provides to the resolvers all the required cryptographic material (keys, signatures, etc...) to validate that the records were not modified. The deployment of DNSSEC is progressing slowly, e.g., in 2017 [6] found that only 12% of the resolvers validate DNSSEC. Our measurements showed a slight increase in DNSSEC adoption; we provide the results in Section 4. In addition, deploying DNSSEC was shown to be cumbersome and error-prone [7, 36].

An important research direction in DNSSEC is cryptographic algorithm agility, which was initiated by [42] who explored the changes needed to deploy new algorithms. Recently [29] studied the lifetime of algorithms for DNSSEC using data from 6.7M signed domains. They found that creating standards for new algorithms or deprecating insecure algorithms takes years. Existing work on algorithm agility focuses mostly on measuring support of algorithms on DNS, or explores addition of new algorithms or removal of insecure ones. In this work we perform the first research on the security considerations of algorithm agility and the possible implications of new algorithms on the security of DNS.

To avoid failures at the resolvers when errors occur during DNSSEC validation Negative Trust Anchors (NTAs) were introduced [27, 33]. NTAs define domains where DNSSEC validation should be turned off. NTAs are not effective against our attacks and typically can be used to support private DNS subtrees that are not referenced from the Internet.

In this work we focus on attack vectors that allow to downgrade DNSSEC: disable DNSSEC validation at the resolvers. DNSSEC-downgrade possibility is considered in [RFC6840] which says that bogus records can be treated as unsigned, leading to downgrade, hence defeating the purpose of DNSSEC. A preliminary poster report of this research raised the question of risks introduced by new algorithms [9]. After we initiated work on our project, a blog post reported a related issue in Google DNS[4] in March 2022.

3 DNSSEC Overview

DNS Security Extensions (DNSSEC) [RFC4033-4035] protects DNS resource records (RRs) against unauthorized manipulations. To enjoy the security guarantees domain owners need to sign the records in their zonefiles with digital signatures and resolvers need to validate the DNS records against the signatures, and discard records with invalid signatures.

DNSSEC records. The cryptographic keys are stored in DNSKEY RRs in a zonefile. These keys are used to validate all the record sets in the zone file. The corresponding signatures are stored in RRSIG records. The keys in the DNSKEY records can be authenticated with the DS delegation records in the zonefile of the parent domain. Each DS RR contains a hash of the corresponding DNSKEY RR of the child domain along with the algorithms in that DNSKEY. For each algorithm supported by the child zone, there should be a corresponding signature in the RRSIG record. When a child is not signed the resolver should check for presence of the proof of negative existence for DS records of the child (typically with a NSEC3 RR). When a child and its parent domains are signed, the parent domain should contain one or multiple DS RRs for the child zone. This enables resolvers to establish a chain of trust to the trust anchor - the key of the root. We illustrate the chain of trust between the root zone "." and the `example.org.` domain in Appendix, Figure 8. Zones are depicted as large cornered boxes. Rounded boxes represent RRsets, while the individual records in a set are given as small cornered boxes and RRSIG records are symbolized by ovals. Signature algorithms are represented by their numbers with a subscript a. Conversely, DS digest types are listed as numbers with a subscript dt. E.g., $2_{dt} \rightarrow 13_a$ specifies a DS record with digest type 2 (SHA256) linking to a DNSKEY record for signature algorithm 13 (ECDSAP256SHA256). DS digest types numbers are specified by [RFC8624]. The DNSKEY of "." signs the DS of `org.`, which contains a hash of the DNSKEY of `org.`, which in turn signs the DS of `example.org.`. That DS record points to the DNSKEY of `example.org.` which signs the RRsets, e.g., `www.example.org.`, in `example.org.`.

Cryptographic algorithms. DNSSEC implementations support a number of cryptographic signing algorithms

[4] https://www.sidnlabs.nl/en/news-and-blogs/a-lock-with-many-keys-spoofing-dnssec-signed-domains-in-8-8-8-8

[RFC8624], and each zone can be signed with one or multiple algorithms. Currently (December 2022) most zones are signed with a single algorithm. In our dataset 99.14% of the domain are signed with one algorithm. Signing the zones with multiple algorithms increases the sizes of DNS responses leading to interoperability problems with intermediate network devices and even exposing to Denial of Service (DoS) attacks. Validation of records signed with multiple ciphers also introduces load on DNS resolvers. Therefore, multiple algorithms are not used to enhance security but are typically used when domains transition to new cryptographic ciphers.

Nameservers. In addition to DNS records, the nameservers must also serve the corresponding DNSSEC records (DNSKEY, RRSIG and DS records) for all the algorithms that they support. This is needed to validate the signatures, as well as to establish a chain of trust to the trust anchor.

Resolvers. In order to establish integrity and authenticity of the DNS records in responses, the resolvers need to validate the RRSIG records over the DS records provided by the parent domain and to use the DS records to establish a chain of trust to the DNSKEY records in the target domain. The keys in the DNSKEY records are used to validate the RRSIG records over the DNS records in the zonefile in the target domain. If the validation of the DS records or the DNSKEY records or the signed DNS records fails, resolvers need to return `SERVFAIL` error message.

Validation of unsupported cryptographic algorithms. When handling zones signed with unknown algorithms signalled in DS RRs, [RFC4035] recommends that the resolvers treat such zones as insecure, i.e., disregard any authenticated DS RRs with unknown or unsupported DNSKEY algorithms. [RFC6840] recommends that DS records with unknown hash digest algorithms should be handled similarly.

When a zone is signed with multiple algorithms, the resolvers needs to validate all the signatures, each time using the corresponding algorithm. When a resolver does not support any of the algorithms it should treat the zone as insecure. Unsupported algorithms are ignored and are not used for validation. An interesting question is how the resolvers should behave when only some of the algorithms are unsupported. Another problem is that the DNSSEC standard does not provide details for handling bogus RRsets. In case of validation state bogus the specific behaviour of resolvers depends on the choices made by the developers. Some implementations return the unauthenticated records to the calling applications while others return a `SERVFAIL` response code instead. The variations in the interpretation of the standard and choices made by developers and operators indicate the lack of understanding and the lack of consensus on best practices.

4 Dataset

In this section we collect our datasets of DNS resolvers and domains that support DNSSEC. For our research we also collect the algorithms that are used in the signed domains. Throughout our work we run experiments and attacks using these datasets of DNSSEC validating resolvers and DNSSEC signed domains. We refer to the algorithms throughout this work by the encoding according to IANA[5]. For convenience we provide the list in Table 2.

	Signature Algorithm							
	RSA				ECDSA		EdDSA	
	5	7	8	10	13	14	15	16
Top-Level Domains	2.12%	1.98%	89.67%	2.34%	4.62%	0.07%	0.00%†	0.00%†
Tranco Top 1M	1.84%	3.19%	33.18%	1.74%	60.07%	0.86%	0.06%	0.02%
BIND 9.11.3	●	●	●	●	●	●	○	○
Knot Resolver 5.3.2	●	●	●	●	●	●	○	○
PowerDNS 4.6.0	●	●	●	●	●	●	●	●
Unbound 1.6.7	●	●	●	●	●	●	●	○
Windows Server 2012	●	●	●	●	●	●	○	○
Windows Server 2012 R2	●	●	●	●	●	●	○	○
Windows Server 2016	●	●	●	●	●	●	○	○
Windows Server 2019	●	●	●	●	●	●	○	○
Windows Server 2022	●	●	●	●	●	●	○	○
CZ.NIC ODVR (Knot)	●	●	●	●	●	●	●	●
Cisco (OpenDNS)	●	●	●	●	●	●	●	●
Cloudflare 1.1.1.1	●	●	●	●	●	●	●	○
Google Public DNS	●	●	●	●	●	●	●	○
Neustar FreeRecursive	●	●	●	●	●	●	●	●
Norton ConnectSafe	●	●	●	●	●	●	●	●
Oracle Dyn	●	●	●	●	●	●	○	○
Verisign	●	●	●	●	●	●	●	●
Adnet Resolvers	99.64%	99.57%	99.71%	99.71%	97.49%	97.42%	70.83%	33.93%
Open Resolvers	99.92%	99.96%	99.96%	100.00%	98.38%	98.38%	85.09%	21.78%

Table 1: Signature Algorithms: Presence in Signed Domains and Support in Resolvers. ●marks support. †exactly zero

Number	Description	Reference
5	RSASHA1	[RFC3110]
7	RSASHA1-NSEC3-SHA1	[RFC5155]
8	RSASHA256	[RFC5702]
10	RSASHA512	[RFC5702]
13	ECDSAP256SHA256	[RFC6605]
14	ECDSAP384SHA384	[RFC6605]
15	ED25519	[RFC8080]
16	ED448	[RFC8080]

Table 2: DNSSEC algorithm numbers we use in our work.

4.1 DNSSEC-validating resolvers

We collect the following datasets of resolvers: (1) popular DNS resolver implementations, (2) popular DNS resolver operators, (3) DNS resolvers used by the web browsers which we collected with an ad network (ad-net), (4) open DNS resolvers, (5) resolvers used by the routing infrastructure for collecting the RPKI objects from the RPKI publication points. A summary of our resolvers' datasets and their support of DNSSEC algorithms is listed in Table 1. Most of the public DNS resolvers and DNS software implementations do not support algorithm 16 (ED448) with a gradually increasing support of algorithm 15 (ED25519).

Popular resolver software. We use nine instances of resolver software in popular versions. These instances cover the market of publicly available resolver software solutions (excluding commercial software) that validate DNSSEC and offer maintenance[6].

DNS resolver services. We use the list of open DNS resolver services at `https://www.publicdns.xyz`. Public

[5] https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml

[6] `https://en.wikipedia.org/wiki/Comparison_of_DNS_server_software`

DNS services are used by many different applications and systems, ranging from web clients, email servers to resolvers of Internet Service Providers (ISPs). In fact, we found that the components in RPKI, the routing security infrastructure, in large part also use public DNS providers. Our measurements in 2021 reveal that 53% of the relying parties[7] (the client side of the RPKI) use public DNS resolvers: 46% use google public DNS and 7% use Cloudflare, both validate DNSSEC.

Open DNS resolvers. This dataset contains publicly accessible resolvers collected via an Internet-wide IPv4 scan for port 53. We obtain a list of 1.8M hosts[8]. We select those hosts that resolve a name in our domain and support DNSSEC, validating a minimum subset of ciphers that correspond to any of the tested algorithms. Our dataset does not contain any hosts that do not respond to our queries or do not resolve a test name in our domain. In effect, the dataset contains 8,829 resolvers, of which 2401 (27.19%) validate DNSSEC.

Resolvers used by web clients. For collecting resolvers of web clients we deployed an advertisement network (ad-net). We distributed our test web page as a pop-under to 35,050 users around the globe, from which 8,977 had the ad page open long enough to have their resolvers fully investigated and 2,785 (31.02%) used validating resolvers. When the investigation web page is loaded in the pop-under, the web browser executes our test script, which instructs it to include a number of images in the document, via URLs containing our test domains. We associate the web clients with the DNS resolvers that they use via a client-specific random token transmitted as a query parameter in each request for an image. We analyze the individual tests by presence of the corresponding web requests. Using the ad-net we identify resolver queries, which originate from 1308 unique, globally distributed IP prefixes. The setup and the study steps are illustrated in Figure 2.

The distribution of the algorithms supported by the DNS resolvers we study using an ad-net and open DNS resolvers are plotted in Figure 1. Most of the validating resolvers, 37.06%, support seven DNSSEC algorithms; 33.68% support eight algorithms, and 26.68% support six algorithms.

[7] Relying party software retrieve the RPKI objects from the public RPKI repositories.

[8] https://opendata.rapid7.com/sonar.udp/

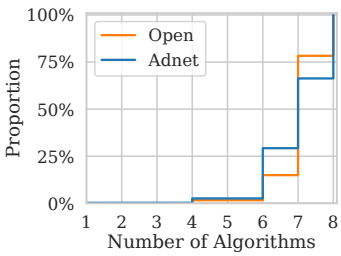

Figure 1: Algorithms supported by open and web-client resolvers.

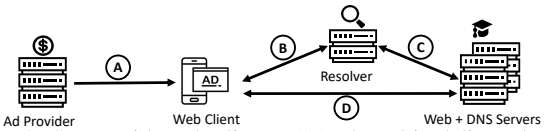

Figure 2: Setup with web clients: (A) The ad is delivered to the browser by the ad provider and (B) causes the client to send queries to the local DNS resolver. (C) The local resolver issues queries to our nameservers and returns the requested records to the client. (D) The client downloads the resource from the web server.

4.2 DNSSEC-signed domains

Our dataset of domains contains 1M-top Tranco domains and Top Level Domains (TLDs). DNSSEC is currently deployed on the DNS root zone using algorithm 8 (RSASHA256). We measure DNSSEC support and algorithms on the list of TLDs from IANA[9] and on 1M-Tranco list[10] of popular domain names. During the measurements we send queries from our client for DS, DNSKEY, and SOA records and obtain the corresponding RRSIG records.

Top level domains. Out of 1,487 TLDs, 1,365 (91.80%) have a DS record at the root zone. Out of the DNSSEC-signed domains, 171 (12.53%) are signed with one cipher (configuration of algorithm and key size), 1,194 (87.47%) are signed with multiple ciphers (3 ciphers max). Measurements of DNSSEC on root and TLDs were done in previous works, e.g., [36], where they also showed that many of the deployments are vulnerable due to reuse of keys.

Second level domains. As a list of second level domains we use the 1M-Tranco list of domains, out of which 5.25% are DNSSEC signed and 35,576 (3.56%) are signed and also have a DS record at the parent. Of those, 24,366 (68.49%) are signed with one cipher, 11,210 (31.51%) with multiple ciphers (with max 7 ciphers).

The distribution of the algorithms (5, 7, 8, 10, 13, 14, 15, 16) are plotted in Figure 4: most domains are signed with a subset of algorithms 7, 8 and 13. The number of different ciphers that the domains are signed with are plotted in Figure 5, with the maximal number of different ciphers being 7.

[9] https://data.iana.org/TLD/tlds-alpha-by-domain.txt

[10] https://tranco-list.eu/

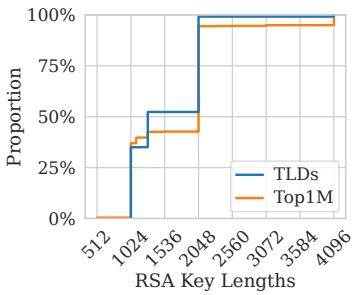

Figure 3: Key lengths in DNSKEYs of 1M-Tranco and TLDs.

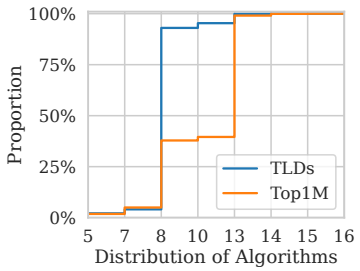

Figure 4: Algorithms in DNSKEY in 1M-Tranco and TLDs.

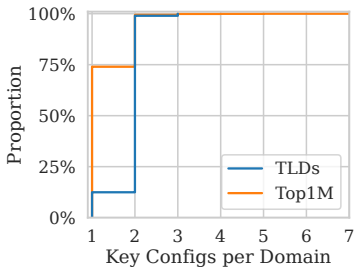

Figure 5: Number of ciphers in DNSKEY Records.

5 DNSSEC-Downgrade Attacks

In this section we develop methodologies for downgrading validation DNSSEC-supporting resolvers for records in signed domains. We first develop the attack vectors for manipulating the responses from signed domains, then describe the zone file configurations we used in our tests, and finally describe the steps of the complete attack. Our attacks cause the resolvers to accept fake DNS records with invalid signatures, without validating the signatures.

5.1 DNSSEC manipulation methodologies

We define the configurations of the victim zone and the methodologies for manipulating the DNSSEC records. We then analyze the DNSSEC standards to understand the factors that allow manipulation of the zone files, leading to disabled DNSSEC validation on the resolvers.

Attack vectors. The attack proceeds by stripping off the RRSIG or DNSKEY records or manipulating the algorithm number in the signature records. The setup is related to the set of algorithms with which the victim zone is signed and to the set of algorithms on the target resolver. We consider the cases when either all or some of the algorithms are supported by the resolver. Our measurements indicate that about 99.25% of the signed zones in our dataset use exactly one signature algorithm and less than 1% use two signature algorithms. We evaluate methods for disabling DNSSEC on single- and dual-signature-algorithm zones via following attack vectors:

```
(a) Strip the RRSIG over the target DNS RRset
(b) Strip the RRSIG over the DNSKEY RRset
(c) Strip the DNSKEY RRset
(d) Rewrite the AlgorithmNumber in the RRSIG
```

The goal of the attack vectors (a), (b) and (c) is to prematurely terminate the chain of trust at the different levels in the validation path. On dual-algorithm zones, attack vectors (a)-(c) will force the resolver along a validation path, which features unsupported cryptographic algorithms. The purpose of attack vector (d) is to interrupt the validation path between the RRSIG and the DNSKEY it belongs to.

Single-signature-algorithm. To carry out tests in this configuration, we set up zones with algorithm 8 (RSASHA256) signatures, as well as DS digest type 2 (SHA256), chosen for widest resolver support. We created illustrations of the attack vectors for each of our zone file configurations using DNSViz, allowing interested readers to access them online: (a)[11], (b)[12], (c)[13] and (d)[14].

Dual-signature-algorithm. In our setup with a dual-signature-algorithm we set up a number of zone files, each signed with two signature algorithms: one supported and one unsupported. Such a configuration includes zones which decide to improve security and adopt a cutting-edge algorithm, or zones that are in the process of rolling over to a new algorithm, that is not supported by a vulnerable resolver. Based on our measurements we consider the signature algorithms {5, 7, 8, 10, 13, 14} as 'supported' and {15, 16} as 'unsupported'. We use algorithms 5 (RSASHA1) and 8 (RSASHA256) as 'supported' to sign our zone files and algorithms 16 (ED448) as well as 18 (unallocated) as 'unsupported' algorithms to sign our dual-signature-algorithm zone file. We created illustrations of the DNSSEC algorithms for each of our dual-signature-algorithm zone file configuration with the corresponding attack vector online: (a)[15], (b)[16], (c)[17] and (d)[18].

Our experimental evaluations show that attack (a) applies to Cloudflare and Google public DNS; 7.41% of open resolvers and 4.96% of ad-net resolvers (cf. 61.06% of open resolvers and 32.35% of ad-net resolvers in 2021). We find that (b) applies to Windows Server DNS and <1% of ad-net and open resolvers (for both 2022 and 2021). Attack vector (c) applies to Windows Server DNS, OpenDNS/Cisco Umbrella DNS, as well as less than 1% of open resolvers and of ad-net resolvers (cf. 4.04% of open resolvers and 1.87% ad-net in 2021).

Factors allowing the attack vectors. We first explain the inherent issues in DNSSEC specification that allow our attacks and then list the main issues in the standard recommendations of DNSSEC validation that expose to vulnerabilities.

First, manipulation of the RRSIG records is possible since the RRSIG records themselves are not protected. If the attacker manipulates the RRSIG record in specific ways, e.g.,

[11] https://www.dnssec-downgrade.net/v/i_sigt_strp_solo
[12] https://www.dnssec-downgrade.net/v/i_sigk_strp_solo
[13] https://www.dnssec-downgrade.net/v/i_key_strp_solo
[14] https://www.dnssec-downgrade.net/v/i_sig_rw_solo
[15] https://www.dnssec-downgrade.net/v/i_sigt_strp_dual
[16] https://www.dnssec-downgrade.net/v/i_sigk_strp_dual
[17] https://www.dnssec-downgrade.net/v/i_key_strp_dual
[18] https://www.dnssec-downgrade.net/v/i_sig_rw_dual

rewrites the 'Algorithm' field from an 8 (RSASHA256) to a 13 (ECDSAP256SHA256), the validator takes the result of that manipulation for granted. In the given example, the validator might look for a DNSKEY of algorithm 13 (and matching KeyTag), would not find any such key, and - abiding by [RFC6840] (Section 5.12) - would then ignore that RRSIG and might raise a validation failure if the RRset in question cannot be authenticated by other means. In contrast, say, Google DNS skipped validation instead, as shown by one of its vulnerabilities. It is also not possible to protect the RRSIG records using other RRSIG records since this is (reasonably) prohibited by [RFC4035], Section 2.2 ("An RRSIG RR itself MUST NOT be signed, as signing an RRSIG RR would add no value and would create an infinite loop in the signing process.").

Second, stripping records is possible because the proofs of non-existence of DNS records do not cover some properties that we exploit in our attacks. The authenticated denial of existence scheme built by means of NSEC-type records (e.g., NSEC3) is sufficient to prove existence and absence of record types at given DNS names. It does not, however, cover presence/absence of records with the requirement of specific other record properties; e.g., it does not cover the signature algorithm types of corresponding DNSSEC records: NSEC can be used to prove that there is a DNSKEY record at the name "example.org.", but it cannot be used to prove that there is a DNSKEY record specifically for signature algorithm 13 at "example.org". Conversely, if there exists at least one DNSKEY record (of any algorithm) at that name, NSEC cannot prove or disprove non-existence of a DNSKEY for any specific algorithm.

To prevent stripping attacks, the presence of DNSSEC records of specific signature algorithm is protected in [RFC4035] (Section-2.2), by setting expectations for resolvers through defining zone requirements. The standard [RFC4035] says: "There MUST be an RRSIG for each RRset using at least one DNSKEY of each algorithm in the zone apex DNSKEY RRset. The apex DNSKEY RRset itself MUST be signed by each algorithm appearing in the DS RRset located at the delegating parent (if any)." This translates into the following rules: (1) if there exists a DS record for a DNSKEY with signature algorithm 'a' at 'example.org', then there must be at least one such DNSKEY record for signature algorithm 'a' at 'example.org'. (2) if there exists a DNSKEY record with signature algorithm 'a' at 'example.org', then, for each RRset in the in the zone 'example.org', there must be at least one RRSIG of signature algorithm 'a'.

This algorithm presence requirement from [RFC4035] is however then relaxed by [RFC6840]. This is one of the key DNSSEC specification issues which expose to our attack vectors, leading to DNSSEC validation vulnerabilities. ("This requirement applies to servers, not validators.") to support algorithm roll-overs over large zones, i.e., to be robust against zones which do not at any point in time feature RRSIGs of both algorithms during an algorithm roll-over. Some very large zones appear to do this, and in doing so, they deliberately hurt the specification. [RFC6840] says "Validators SHOULD accept any single valid path. They SHOULD NOT insist that all algorithms signaled in the DS RRset work, and they MUST NOT insist that all algorithms signaled in the DNSKEY RRset work." [19] This statement from the RFC is arguably geared towards completeness checks, as shown by the next sentence: "A validator MAY have a configuration option to perform a signature completeness test to support troubleshooting.". That purpose does not appear to be obvious and the "MUST NOT insist that all algorithms signaled in the DNSKEY RRset work" should have been explicitly augmented with a ", provided at least one supported working algorithm exists."

5.2 Cache poisoning methodologies

Merely downgrading DNSSEC validation does not constitute a meaningful attack. We therefore describe combinations of DNSSEC downgrade attack vectors (a)-(d) with cache poisoning attacks for injecting fake DNS records. Which fake records are accepted by a given resolver and under what conditions, depends on the specific caching policies implemented by the resolvers. These caching policies include the rank that caching implementations assign to different responses types, e.g., referral or Answer type responses, as well as the considerations of resolvers for using the stored records, e.g., stored keys with or without re-evaluating the whole chain of trust. Caching policies for injections of spoofed records were explored in [21, 37].

Hijack secure DNSSEC delegation. The goal of the adversary is to inject a malicious DNSKEY which it can use to sign fake records in a victim domain. In this example we assume that the victim resolver does not support algorithm 16 and supports algorithm 8. To carry out the attack the adversary needs to manipulate a referral type response sent by the nameserver to the DNS resolver. The relevant part of the referral response contains a DS record that corresponds to the genuine DNSKEY of the victim domain, which the adversary does not control (for instance f3...cc) and two RRSIG records:

```
IN DS 29449 13 2 f34135...eecc
IN DS 29449 13 4 8e1ec0.....180f
IN RRSIG DS 8
IN RRSIG DS 16
```

To do that, the adversary strips off (using attack vector (a) from Section 5.1) the RRSIG with the supported algorithm 8 and replaces the digest of the DS record with:

```
IN DS 5342 13 2 bd638a.....4303
IN RRSIG DS 16 // invalid
```

[19] One of the DNS vendors used this sentence to argue for the RFC-compliance of their product when we did the disclosure with them.

Now the response contains only the unsupported RRSIG 16 for validation. The signature in the RRSIG 16 is invalid since it does not fit to the modified DS record. However, since algorithm 16 is not supported, the resolver does not perform DNSSEC validation. In this case the vulnerable resolver should return `SERVFAIL`. If the resolver instead accepts and caches the new DS record that corresponds to the key of the adversary, the adversary would be able to inject any record of its choice. Since the manipulated DS corresponds to adversaries own key-pair the adversary can manipulate any records for that domain and sign them with a private key that corresponds to the value in DNSKEY. The DNSSEC validation of the resolver would be successful. As a result, the adversary could not only hijack the entire DNSSEC-secure DNS tree but would also be able to forge records in that tree with correct DNSSEC signatures. Responses from the real domain would in that case fail validation since they use a different DNSKEY record that not present in resolver's cache.

Disable secure DNSSEC delegation. To disable DNSSEC validation in a DNS sub-tree, in addition to manipulating the algorithm number in an RRSIG to an unsupported algorithm in referral responses, the adversary also changes the DS to contain an unsupported algorithm number. Assume the nameserver returns a referral with the following records:

```
IN DS 5342 8 2 f34135.....eecc
IN DS 5342 8 4 8e1ec0.....180f
IN RRSIG DS 13
IN RRSIG DS 16
```

The adversary strips off the supported RRSIG 13 (attack vector (a), Section 5.1) and changes the supported algorithm number 8 in DS to an unsupported algorithm 16.

```
IN DS 5342 16 2 f34135.....eecc
IN DS 5342 16 4 8e1ec0.....180f
IN RRSIG DS 16 // invalid
```

If the resolver accepts and caches the records, the validation in that subdomain is not performed. Consequently, the entire subtree under the corresponding DS records is treated by the vulnerable resolver as insecure. This would allow the adversary to replace any record in any subdomain, without adjusting the signatures. The victim resolver also accepts responses from genuine nameservers without validating them.

Hijack secure domain. The adversary hijacks a domain by injecting a malicious nameserver. The adversary can manipulate the IP addresses in a referral DNS response. In referrals the glue records are not signed, hence this step does not require a manipulation of the RRSIG to an unknown algorithm. A subsequent *authoritative* response from the domain contains signed NS records, which have higher caching rank than the unsigned NS records in the referral. Therefore, the adversary would also need to perform another step to downgrade the DNSSEC validation protecting the NS RRset or/and A RRset and inject malicious NS/A records. In all subsequent responses from the adversarial nameservers, the adversary has

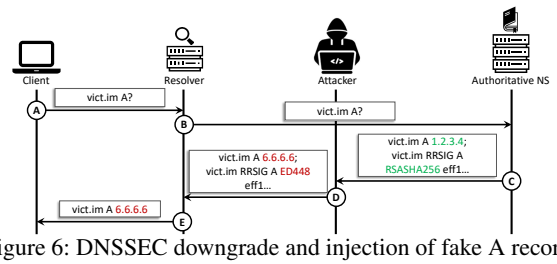

Figure 6: DNSSEC downgrade and injection of fake A record.

to apply one of the attack vectors (a)-(d) to continue disabling DNSSEC validation.

Manipulate records in Answer responses. To manipulate genuine records in a DNS response or to inject a fake record to the response the adversary has to employ one of the attack vectors (a)-(d). Depending on the attack vector, the adversary can create a new DNSKEY record for a domain or hijacking a resource by manipulating the algorithm and changing the hostname or an IP address.

5.3 Complete attack

We combine the attack vectors for manipulating the DNSSEC records into a full fledged attack for downgrading DNSSEC validation in a victim resolver. The attack is illustrated in Figure 6, which demonstrates attack vector (d). Adversary causes a recursive resolver to issue a query. There are different ways to do that, in an example in Figure 6 we illustrate query triggering using web clients that download our object via an ad-network that we deployed. The client sends a query to the recursive resolver (step (A)), which in turn sends it to the authoritative DNS server of the victim domain (step (B)). The response of the nameserver is in step (D) changed by an on-path attacker, according to one of the attack vectors (a)-(d) defined in Section 5.1. In addition, the adversary injects a malicious DNS record. The new record would not pass DNSSEC validation, since the modified record does not match to the digest in the signature. However, the DNSSEC validation at the resolver was disabled for that answer. Therefore, the resolver caches the malicious records that poison its cache, despite the fact that the signature is invalid.

6 Evaluations

In this section we describe evaluations of the downgrade attacks. We first provide details of the infrastructure that we set up for running the evaluations and Internet wide measurements. Then we describe the evaluations that we run on our datasets of resolvers.

6.1 Setup

The setup contains the test domains that we control, authoritative nameservers (based on Knot implementation [30]) in

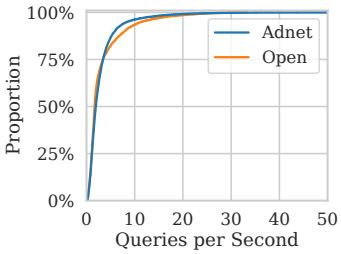

Figure 7: Query rate: 3.25 avg. open resolvers, and 3.00 ad-net.

	Any Domain		Single Alg. Domains		Multiple Alg. Domains					
Top-Level Domains	100.00%		99.19%		0.00%		0.00%		0.00%	
Tranco Top 1M	100.00%		99.14%		0.04%		0.04%		0.04%	
attack vectors (§5.1)	i_any: (a)-(d)		i_sig_rw: (d)		i_sigt_strp: (a)		i_sigk_strp: (b)		i_key_strp: (c)	
	2021	2022	2021	2022	2021	2022	2021	2022	2021	2022
BIND 9.11.3	○	○	○	○	○	○	○	○	○	○
Knot Resolver 5.3.2	○	○	○	○	○	○	○	○	○	○
PowerDNS 4.6.0	○	○	○	○	○	○	○	○	○	○
Unbound 1.6.7	○	○	○	○	○	○	○	○	○	○
Windows Server 2012	●	●	○	○	○	○	●	●	●	●
Windows Server 2012 R2	●	●	○	○	○	○	●	●	●	●
Windows Server 2016	●	●	○	○	○	○	●	●	●	●
Windows Server 2019	●	●	○	○	○	○	●	●	●	●
Windows Server 2022	●	●	○	○	○	○	●	●	●	●
CZ.NIC ODVR (Knot)	○	○	○	○	○	○	○	○	○	○
Cisco (OpenDNS)	●	○	○	○	○	○	○	○	●	○
Cloudflare 1.1.1.1	●	●	○	○	●	●	○	○	○	○
Google Public DNS	●	○	●	○	●	○	○	○	○	○
Neustar FreeRecursive	○	○	○	○	○	○	○	○	○	○
Norton ConnectSafe	○	○	○	○	○	○	○	○	○	○
Oracle Dyn	○	○	○	○	○	○	○	○	○	○
Verisign	○	○	○	○	○	○	○	○	○	○
Adnet Resolvers	34.18%	5.03%	27.58%	0.18%	32.35%	4.96%	0.11%	0.11%	1.87%	0.11%
Open Resolvers	65.10%	7.50%	53.52%	0.04%	61.06%	7.41%	0.17%	0.17%	4.04%	0.08%

Table 3: Downgrade vulnerabilities in resolvers and victim domains.●marks vulnerability. Domain percentages are relative to the number of signed domains with DS records at the parent.

our test domains, a Man-in-the-Middle (MitM) proxy and a client component. The setup corresponds to the illustration in Figure 6, with "attacker" being our MitM proxy.

Domains and nameservers. The nameservers of each domain are configured to use the zonefiles described in Section 5.1. The nameservers serve records in each of the zones, signed with all the algorithms that can be found in the corresponding DS records.

Reverse MitM proxy. The reverse MitM proxy is developed in python and acts as an on-path adversary. It manipulates the records in the DNS responses according to the attack vectors (a)-(d) defined in Section 5.1.The MitM proxy replaces the algorithms' numbers in the signatures or removes the records from the responses.

Client. We use a client side component to cause the target resolvers to issue queries to our domains. The setup and functionality of the client side component can be direct and indirect, depending on the access to DNS. The former is when the queries to the resolver can be sent directly, while the latter applies when the evaluation is performed indirectly via our script in clients' browsers.

Direct tests. A client side component is simple, it is located on our measurement infrastructure and triggers queries to the target DNS resolvers directly. We use this setup to test the publicly available implementations of DNS, public DNS providers and open DNS resolvers.

Indirect tests. A script running in client's browser issues the DNS queries, which are passed on by the stub resolver on its device to its local recursive DNS resolver. We use this setup in our ad-net study of web clients. The client component tests the attack vectors (a)-(d) each time triggering requests to our test domains. The requests encode different tests. Upon each request, the nameserver decodes the test from the query name sent by the DNS resolver, and sends a response. In our evaluations we issue on average 3 queries per second in the ad-net study and 3.25 in the study of open resolvers; the query rate is plotted in Figure 7.

6.2 Results

We list the results of our evaluations of the downgrade attacks against the resolvers in our datasets in Table 3. The upper part lists the domains (TLDs and 1M-top Tranco domains) that have DNSSEC configurations, which cause the resolvers not to apply DNSSEC validation, exposing the vulnerable resolvers to cache poisoning attacks.

The most recently standardized DNSSEC algorithms are not yet widely supported by resolvers and are thus rarely used to sign zones. Combinations of different algorithm numbers are usually found in domains that are in transition from one algorithm to a new one, which is not too common. The rounded zero percentages in Table 3 conform to absolute values of zero, except where noted otherwise. In addition to domains with DNSSEC configurations that disable DNSSEC in vulnerable resolvers, we use our downgrade methodologies (developed in Section 5) to disable DNSSEC irrespective of DNSSEC configurations in domains.

The next category in Table 3 is a list of DNS software which we found to be vulnerable. The different Windows server versions were installed in a lab setup, and evaluated against the domains in our dataset. The four additional resolver implementations in Table 3 were tested in the same lab environment and were not vulnerable. Then, the table lists a category of public DNS operators. The last two categories are measurements of the downgrade attacks in open DNS resolvers and in DNS resolvers we measured using an ad-net. Percentages are relative to the number of validating resolvers, and to the number of clients that use validating resolvers.

Our evaluations show no difference in resolver behaviour when testing an unsupported algorithm, does not matter if its algorithm number is assigned or is unassigned. Therefore, in the rest we focus on evaluations with unsupported assigned algorithm numbers.

Public DNS operators. We evaluate our attacks against the public DNS operators in our dataset and find Cloudflare, Google Public DNS and OpenDNS to be vulnerable to our attacks. In addition, we find that all the three exhibit non-RFC compliant validation behaviour.

• Cloudflare resolver is vulnerable to downgrade attacks with zones that are signed with any arbitrary unsupported algorithm. When Cloudflare DNS resolvers encounter an unsupported as well as one supported algorithm in the validation path, beginning with one of the DS records or DNSKEY, and

in RRSIG, an attacker can inject fake records and employ attack vector (a) to coerce Cloudflare along the unsupported validation path. Despite the fact that DNSSEC validation is invalid, Cloudflare does not of return SERVFAIL. Instead, unauthenticated records are returned. While acceptable when the DS RRset does not contain any supported algorithms, this behavior also occurs when there exists a validation path with supported DS algorithms, DNSKEYs and RRSIGs. The standard recommends returning SERVFAIL in those cases, however, Cloudflare does not insist on any supported signatures to exist when unsupported DS/DNSKEY records are present.

Since the beginning of our measurements in 2021, Cloudflare supports all tested algorithms except ED448 (16) (see Table 1). In our proof of concept we use algorithm 16.

• OpenDNS is vulnerable to downgrade attacks with zones, that have at least one DS record conforming to an unsupported algorithm. The attacker proceeds by injecting fake records and stripping off the supported DNSKEY as well as all its signatures from responses to the victim resolver, thereby leaving the supported DS record at the parent zone as only indication, that RRSIGs conforming to supported signature algorithms should exist in the victim zone. By doing so, the attacker tricks OpenDNS into believing that no supported signing information exists in the zone, forcing it along the unsupported validation path. Being vulnerable, OpenDNS downgrades security on the records from the target zone and delivers fake records to the client for which the signatures are invalid. This behavior is non-compliant with [RFC4035], Section 5, which says "The absence of DNSSEC data in a response MUST NOT by itself be taken as an indication that no authentication information exists.". Instead, OpenDNS should return SERVFAIL for any request to the victim zone and all its descendants.

At the time of our evaluations in 2022 OpenDNS does support all signature algorithms, which are allowed for signing zones [RFC8624]. For our proof of concept, we use DNSKEYs and RRSIGs designated with the unassigned algorithm number 18. We create valid records for algorithm ED448 (16), re-label them and adjust DS hashes, key tags and other record fields which depend on the algorithm number. This allows us to evaluate resolver behavior on test zones conforming to arbitrary algorithm numbers.

• Google Public DNS (8.8.8.8) was found vulnerable to both attack vector (a) as well as attack vector (d). The attack scenario employing (a) proceeds similarly to Cloudlfare. Additionally to (a), Google has also been found vulnerable to (d). This attack affects any domain on the Internet, including TLDs. We next describe the procedure on a zone using a single signature algorithm.

The attacker injects fake records into the response and rewrites the algorithm numbers in the contained RRSIGs to unsupported ones. In our measurements, Google Public DNS (8.8.8.8) supported all algorithms except ED448 (16). In our proof of concept evaluation we use signature algorithm 16, that is not part of the test zone, as indicated by DNSKEY or DS. By rewriting the RRSIG algorithm numbers, we dissociate the authentic RRSIGs covering the modified records from their authentic DNSKEY(s), and, therefore, from the rest of the validation path. To the attacked resolver, the attacked zone looks like there are no RRSIGs conforming to the authenticated DNSKEYs, but only those RRSIGs, which do not conform to any key at all. According to [RFC6840] Section 5.12, these should be ignored: "Validating resolvers MUST disregard RRSIGs in a zone that do not (currently) have a corresponding DNSKEY in the zone. Similarly, a validating resolver MUST disregard RRSIGs with algorithm types that do not exist in the DNSKEY RRset." After ignoring those RRSIGs, in our PoC, there are no RRSIGs left to consider. Resolvers abiding by [RFC4035] would then raise a validation failure, since "The absence of DNSSEC data in a response MUST NOT by itself be taken as an indication that no authentication information exists.". Google Public DNS, however, behaved non-compliant in this case: It instead downgrades the security of the covered records. We locate the non-compliance in [RFC6840], since, as we measured, if no RRSIG exists on the queried RRset at all, then SERVFAIL is returned.

The market shares of the Public DNS operators are reflected by the vulnerability statistics of open resolvers as well as resolvers used by web clients. We use the statistics on market shares to calculate the 2021 vulnerability statistics for all vulnerabilities, which we found and reported over the course of our research. We next describe our findings.

Open DNS resolvers. Table 3 lists the fraction of resolvers that were vulnerable to at least one attack method for the investigated domain configurations. Most resolvers (61.06%) were found vulnerable when using attack vector (a) in Section 5.1, followed by (d) with 53.52%, and (c) with 4.04%. The smallest number of resolvers, 0.17%, were vulnerable to (b).

Resolvers used by web clients. The setup for ad-net study is illustrated in Figure 2 and inspired by [15]. When our script is downloaded by the client it iteratively includes resources (img) from test domains, including a non-DNSSEC-signed domain to signal session finish. The web server logs all the requests and delivers the requested resources. A script then analyzes the logs to check for vulnerabilities to our DNSSEC downgrade attack. The steps of the evaluation are the following: (A) the ad is delivered to the web client, causing it to fetch and execute our test script. (B) The client queries its resolver for names in our test domains. (C) The resolver looks up those names querying our authoritative DNS server, we test vulnerabilities to downgrade attacks. If vulnerable to one of the tests, the resolver successfully answers the query and (D) the client fetches the requested resource from our web server. We derive resolver vulnerability from the existence of the corresponding requests in the web log.

In our ad-net study we covered 1,232 Autonomous Systems (ASes) with publicly routable prefixes. From the covered ASes, our server was accessed by 7.29 clients per AS on aver-

age. The ad-net study spanned 154 countries around the globe, which homed 56.98 clients on average. We observed behavior of resolvers used by 8,977 ad-net clients, out of which 2,785 used validating resolvers. We found 901 (32.35%) users to use resolvers vulnerable to attack vector (a). Out of those, 286 were located in Asia, 157 in the EU and 116 in North America. 1.87% were vulnerable to attack vector (c). 15 of them sent requests from addresses geo-located in Asia, 20 in the EU and 3 in North America. 27.58% of adnet clients used resolvers, which were vulnerable to attack vector (d). The share was 0.11% for attack vector (b).

Our measurement results indicate that a large portion of Internet clients use resolvers that are vulnerable for at least some DNSSEC configuration. While the adversary cannot choose the DNSSEC configuration of the attack target, for a successful attack it suffices that *any* of the DNSSEC configurations in the target zone's ancestors have a configuration vulnerable at the resolver.

7 Recommendations and Challenges

In this section we discuss the key issues that allow our attacks and provide recommendations for updating the DNSSEC standard as well as concrete mitigations to prevent DNSSEC downgrade attacks.

Ensuring standard compliance. DNS developers and public DNS providers should support the validation behaviour recommended in the standard. This would prevent the part of our attacks that exploit non standard behaviour, such as those of Google public DNS, OpenDNS and CloudFlare. Nevertheless, adhering to the standard does not solve all the vulnerabilities. Our analysis of the standard shows that lack of clear behaviour specification for DNSSEC validation outcomes in case of "indeterminate" and "bogus" states introduces a vulnerability which we exploit in our attacks:

"Indeterminate" state: The "indeterminate" security state, which is one of the aspects we exploit in our downgrade attacks, is defined differently in both DNSSEC standards [RFC4033] and [RFC4035]. This is not only confusing but also opens up room for different interpretations by the developers of DNS resolvers. Clear and consistent definitions are imperative for avoiding different interpretations.

"Bogus" state: Our study shows that when all the algorithms are supported resolvers identify the "bogus" cases correctly, e.g., when there is a clear validation failure or some of the DNSSEC records are missing (e.g., no DNSKEY at all). However, the situation changes as soon as multiple algorithms are involved, some of which supported and some not, because then, the "resolvers MUST NOT insist on all algorithms from DNSKEY to work", which opens up room for confusion and misinterpretations. In our evaluations we identify resolvers that do not return `SERVFAIL` responses in cases when multiple algorithms are present and not all of them are supported. For instance, when the DS records contain supported and unsupported algorithms, but the zone is signed only unsupported DNSKEY and RRSIG. Behaviour of some resolvers exposes them to attack vector (c) in §5.1 (i_key_strp in Table 3), since they do not return `SERVFAIL` in those cases. This is possibly caused by [RFC4033] Section 5, which says that bogus state *can* be communicated to the clients via `SERVFAIL`. However, this case should also be bogus since the DS RRset signals that a supported DNSKEY and the corresponding RRSIG should be present at the child.

Developing robust standards. Our research shows that aiming to offer flexibility of implementations may lead to confusion and discrepancy in different interpretations by the developers. On the other hand, over specification is also risky. A recent effort [24] for developing protocols robust to failures, bugs and attacks, discusses the risks of flexibility derived from the architectural principle (Postel's law) of the Internet [RFC1958] "Be strict when sending and tolerant when receiving." and refers to the possibility of differing interpretations [RFC0760]. Namely, the downgrade behaviour is left to the implementers to choose. As a consequence, the robustness principle of DNSSEC implementations exposes to our attacks. On the other hand, [24] considers the importance of deployment over a perfect specification since the protocols benefit greatly from user experience when they are used.

Cryptographic standards often try to solve this problem by clearly, often even formally specifying their assumptions regarding the security of external functions and operations they build and rely on, and formally proving that these assumptions are sufficient for providing their intended security functions and properties. This approach works often well for "small" cryptographic functions. However, proving the security of larger specifications is still a challenge. Another approach for assuring the security of standards is to provide incentives for the security research community for carefully analyzing draft standards, e.g., through competitions and corresponding sessions at top scientific conferences.

Hardfail at "bogus". Trivial solutions, like checking that the algorithm number is within the list of all the standardised algorithm numbers, do not work. New unknown algorithm types may be included in DNSSEC and the DNSSEC records may be provisioned to be used in different ways than they were originally planned. For instance, a recent draft proposes to signal support of DoT using a new DNSKEY algorithm type TBD[20]. As a systematic solution to prevent downgrade attacks we propose to update the DNSSEC specification and impose rigorous checks on algorithms presence and a more rigorous requirement for `SERVFAIL` return codes when an RRset qualifies as "bogus".

Algorithm presence: To prevent our DNSSEC downgrade attacks we recommend that the resolvers insist on algorithm presence - as mandated analogously by [RFC4035] for the zone side. If there exists a supported DS record for

[20] `https://datatracker.ietf.org/doc/html/draft-vandijk-dprive-ds-dot-signal-and-pin-01`

a DNSKEY with supported signature algorithm 'a' at 'example.org', then the resolver must insist on at least one such DNSKEY record for signature algorithm 'a' at 'example.org'. If there exists a DNSKEY record with a supported signature algorithm at 'example.org', then the resolver must insist on least one RRSIG of a supported signature algorithm for each RRset in the in the zone 'example.org'. The rationale behind this: When considering the signature algorithms and DS digest types of a zone, a resolver may only skip validation if all DS records conform to a digest type, which is not supported, or signal a signature algorithm, which is not supported (either one is sufficient for each individual DS digest). Namely, missing support for DS digest types or signature algorithms may result in skipped validation, if and only if there exists no DS record with both a supported DS digest type and supported signaled signature algorithm.

Requirement for return codes during bogus: We recommend to return `SERVFAIL` when validation status is "bogus". This would prevent the attacks without imposing restrictions on the usage of DNSSEC and without limiting the flexibility of deployment of new algorithms. However, enforcing this will face obstacles due to practical considerations which may cause them to be seen as controversial by the community. Insisting on returning `SERVFAIL` may cause failures during key rollover, when a rollover is made to an unknown algorithm and the key that corresponds to a known algorithm is expired. Additionally, returning `SERVFAIL` prevents key prepublication [RFC6781]. Key prepublication is one of the ways for zone-signing key rollover: a new key is added into the DNSKEY RRset and the RRset is re-signed. Signatures created with the old key can be replaced with the new key. Returning `SERVFAIL` causes validation failures in such cases.

Another issue is inconsistency in records between parent and child domains [6, 31, 32, 38]. This can happen, e.g., when the set of the keys in the DS records does not overlap with the keys in the DNSKEY records in the child zone. As a result, the resolver cannot establish a chain of trust to the trust anchor. In our measurements of inconsistencies between the parent and the child domains in October 2021 we find that about 0.27% of popular signed domains have misconfigurations for which DNSSEC validation would fail. Insisting on DNSSEC validation in those cases would break the access to the services in those domains, hence the DNS operators decided to preserve connectivity sacrificing security. Unfortunately, this is exactly the vulnerability that we exploit in our DNSSEC downgrade attacks.

Validate DNSSEC in resolvers. There are tools for validating deployment of DNSSEC on domains, such as DNSViz, which check if the zone is correctly signed, i.e., if signatures are present and chain of trust can be established from the root. Those tools however do not detect the scenarios which make resolvers vulnerable to our downgrade attacks. We develop a tool for evaluating the DNSSEC validation in resolvers in different situations which can be exploited for downgrade attacks: `https://www.dnssec-downgrade.net/`. Vulnerabilities are reported in an output with explanations of vulnerable scenarios and recommendations for countermeasures.

8 Conclusions

Cryptographic algorithm agility in DNSSEC, i.e., the ability to add and remove algorithms, is an important requirement needed to maintain strong security guarantees. Algorithms may be broken, being able to replace vulnerable algorithms with secure ones efficiently and fast is critical [13].

We show that efficient and fast adoption of algorithms also introduces a challenge: how should resolvers react when faced with records signed using new algorithms? What is the correct behaviour with zones that are signed with a number of algorithms, some of which are known? The standard does not provide clear recommendations for resolvers how to handle DNSSEC records with unknown algorithms and how to handle bogus data, but leaves it open for every resolver to make its own decision how to behave in such cases. Although flexibility in standards is important, care should be taken to avoid vulnerable implementations. We discover that this vague specification leads to different validation behaviour in popular DNS resolver implementations, which indicates that there is no consensus on what a correct behaviour should be. The implementers and operators often prefer to ignore DNSSEC all together instead of returning `SERVFAIL` when the records cannot be validated. This decision is sometimes made to preserve connectivity to domains, since there are also benign situations that can lead to `SERVFAIL`, e.g., due misconfigurations, deployment of new algorithms or key rollover. However, we show that this tradeoff also leads to vulnerabilities. If an adversary modifies a DNS response, e.g., by manipulating an RRset signature's algorithm number to an algorithm not listed in the DS RRset, it can cause the RRset to qualify as "bogus" leading to undefined behavior in the specification. In this work we show that resolvers, that handle bogus cases as if the domain were not signed with DNSSEC and the RRset is "insecure", are vulnerable.

Our work calls for a quick action to extend and clarify the standard as well as patch the vulnerabilities to make the cryptographic algorithm agility in DNSSEC secure.

Acknowledgements

This work has been co-funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) SFB 1119.

References

[1] Domain name system security (DNSSEC) algorithm numbers. https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml.

[2] F. Alharbi, J. Chang, Y. Zhou, F. Qian, Z. Qian, and N. Abu-Ghazaleh. Collaborative client-side dns cache poisoning attack. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019.

[3] Dan J. Bernstein. DNS Forgery. http://cr.yp.to/djbdns/forgery.html, November 2002.

[4] Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. Bamboozling certificate authorities with {BGP}. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 833–849, 2018.

[5] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Shulman, and Michael Waidner. Domain Validation++ For MitM-Resilient PKI. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2060–2076. ACM, 2018.

[6] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. A longitudinal, end-to-end view of the {DNSSEC} ecosystem. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 1307–1322, 2017.

[7] Taejoong Chung, Roland van Rijswijk-Deij, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. Understanding the role of registrars in dnssec deployment. In *Proceedings of the 2017 Internet Measurement Conference*, pages 369–383, 2017.

[8] Tianxiang Dai, Haya Shulman, and Michael Waidner. Let's downgrade let's encrypt. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1421–1440, 2021.

[9] Elias Heftrig, Haya Shulman, and Michael Waidner. Poster: The unintended consequences of algorithm agility in DNSSEC. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 3363–3365. ACM, 2022.

[10] Amir Herzberg and Haya Shulman. Security of Patched DNS. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, pages 271–288, 2012.

[11] Amir Herzberg and Haya Shulman. Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In *IEEE CNS 2013. The Conference on Communications and Network Security, Washington, D.C., U.S.* IEEE, October 2013.

[12] Tomas Hlavacek, Philipp Jeitner, Donika Mirdita, Haya Shulman, and Michael Waidner. Stalloris: RPKI Downgrade Attack. In *31th USENIX Security Symposium (USENIX Security 22)*. USENIX Association, August 2022.

[13] Russ Housley. Guidelines for cryptographic algorithm agility and selecting mandatory-to-implement algorithms. Technical report, BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015,< https://www.rfc . . . , 2015.

[14] A. Hubert and R. van Mook. Measures for making dns more resilient against forged answers. RFC 5452, RFC Editor, January 2009.

[15] Geoff Huston. Measuring the end user, 2015.

[16] Dan Kaminsky. It's the End of the Cache As We Know It. Presentation at Blackhat Briefings, 2008.

[17] Amit Klein. Bind 9 dns cache poisoning. *Report, Trusteer, Ltd*, 3, 2007.

[18] Amit Klein. Windows dns server cache poisoning, 2007.

[19] Amit Klein. Cross layer attacks and how to use them (for dns cache poisoning, device tracking and more). *arXiv:2012.07432*, 2020.

[20] Amit Klein, Haya Shulman, and Michael Waidner. Internet-wide study of dns cache injections. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[21] Amit Klein, Haya Shulman, and Michael Waidner. Internet-Wide Study of DNS Cache Injections. In *INFOCOM*, 2017.

[22] Wilson Lian, Eric Rescorla, Hovav Shacham, and Stefan Savage. Measuring the Practical Impact of {DNSSEC} Deployment. In *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, pages 573–588, 2013.

[23] Baojun Liu, Zhou Li, Peiyuan Zong, Chaoyi Lu, Haixin Duan, Ying Liu, Sumayah Alrwais, Xiaofeng Wang, Shuang Hao, Yaoqi Jia, et al. Traffickstop: Detecting and measuring illicit traffic monetization through large-scale dns analysis. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 560–575. IEEE Computer Society, 2019.

[24] Thomson M. and Schinazi D. Maintaining Robust Protocols, June 2023.

[25] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[26] Keyu Man, Xin'an Zhou, and Zhiyun Qian. Dns cache poisoning attack: Resurrections with side channels. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3400–3414, 2021.

[27] Matthijs Mekking. A Story of Unsupported DNSSEC Algorithms, June 2019.

[28] P. Mockapetris. Domain names - implementation and specification. STD 13, RFC Editor, November 1987. http://www.rfc-editor.org/rfc/rfc1035.txt.

[29] Moritz Müller, Willem Toorop, Taejoong Chung, Jelte Jansen, and Roland van Rijswijk-Deij. The reality of algorithm agility: Studying the dnssec algorithm lifecycle. In *Proceedings of the ACM Internet Measurement Conference*, pages 295–308, 2020.

[30] CZ NIC. It's knot dns. 2011.

[31] Eric Osterweil, Michael Ryan, Dan Massey, and Lixia Zhang. Quantifying the operational status of the dnssec deployment. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 231–242, 2008.

[32] Eric Osterweil, Pouyan Fotouhi Tehrani, Thomas C. Schmidt, and Matthias Wählisch. From the beginning: Key transitions in the first 15 years of dnssec, 2021.

[33] Ebersman P., Kumari W., Griffiths C., Livingood J., and Weber R. RFC7646: Definition and Use of DNSSEC Negative Trust Anchors, September 2015.

[34] Craig Partridge and Mark Allman. Ethical considerations in network measurement papers. *Communications of the ACM*, 59(10):58–64, 2016.

[35] Haya Shulman and Michael Waidner. Towards security of internet naming infrastructure. In *European Symposium on Research in Computer Security*, pages 3–22. Springer, 2015.

[36] Haya Shulman and Michael Waidner. One key to sign them all considered vulnerable: Evaluation of {DNSSEC} in the internet. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 131–144, 2017.

[37] Sooel Son and Vitaly Shmatikov. The hitchhiker's guide to dns cache poisoning. In *Security and Privacy in Communication Networks*, pages 466–483. Springer, 2010.

[38] Roland van Rijswijk-Deij, Taejoong Chung, David Choffnes, Alan Mislove, and Willem Toorop. The root canary: Monitoring and measuring the dnssec root key rollover. In *Proceedings of the SIGCOMM Posters and Demos*, pages 63–64. 2017.

[39] Paul Vixie. DNS and BIND security issues. In *Proceedings of the 5th Symposium on UNIX Security*, pages 209–216, Berkeley, CA, USA, jun 1995. USENIX Association.

[40] S. Weiler and D. Blacka. Clarifications and implementation notes for dns security (dnssec). RFC 6840, RFC Editor, February 2013. http://www.rfc-editor.org/rfc/rfc6840.txt.

[41] Wouter CA Wijngaards and Benno J Overeinder. Securing dns: Extending dns servers with a dnssec validator. *IEEE Security & Privacy*, 7(5):36–43, 2009.

[42] Dan York, Ondřej Surỳ, Paul Wouters, and O Gudmundsson. Observations on deploying new dnssec cryptographic algorithms. Technical report, Tech. Rep, 2016.

[43] Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan, and Zhiyun Qian. Poison over troubled forwarders: A cache poisoning attack targeting DNS forwarding devices. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 577–593, 2020.

A Disclosure and Notification

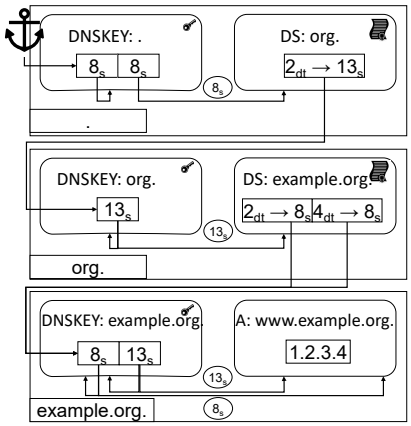

Figure 8: DNSSEC Chain of Trust Schematic.

In this section we provide further information about our disclosure efforts and communication with the affected vendors.

We informed Cloudflare, Microsoft, Google, as well as Cisco about the vulnerabilities in their DNS products. We additionally contacted Nic.fr as well as Verisign and informed them about the new attack vectors we found.

As of January 2023, Google an Cisco fixed the vulnerabilities. Our experimental evaluations in January 2023 show that the issues persist in an up-to-date Windows Server 2019. At the time of writing, Cloudflare's public resolver service remains vulnerable. More so, Cloudflare reportedly[21] sets negative trust anchors (NTAs) on domains "with detected and vetted DNSSEC errors". This further illustrates the delicate balance, which resolver operators strive to find between reliability and security.

[21] https://blog.cloudflare.com/dns-resolver-1-1-1-1/